Interactive Exploration of Versions across Multiple Documents

Chang-Han Jong, Prahalad Rajkumar, Behjat Siddiquie

***Abstract*—Researchers in literature need to compare several versions of one or more poems or articles for investigating their historical and literary significance. Current tools do not adequately support their requirements. We address some of these issues by developing Multiversioner, a tool designed to interactively analyze multiple documents, each consisting of several versions. Additionally, it enables users to search for entities such as words and lines and also facilitates the analysis of frequency patterns of these entities. We have extensively used visualization principles such as use of color-coded highlighting, overview with details on demand and simple controls. Promising feedback has been received from a domain expert. Some of these techniques have applications in other domains.**

***Index Terms*—versions, comparison, multiple documents, literature, visualization**

I. Introduction

THE need to compare two or more documents with each other arises in a variety of situations. Some instances include detection of plagiarism in academic settings, comparing versions of computer programs, and comparing the flow of history in the wiki setting such as wikipedia articles [2]. Lots of research has been devoted to comparing documents with each other, such as [7],[8],[10]. Though there exist several tools such as windiff to visually compare a pair of documents, little work has been done on providing an effective visual interface to facilitate the comparison of several documents. Versioning Machine (http://www.v-machine.org) by Schreibman et all is a web-based interface that provides the facility to view multiple versions of a document, along with the changes across versions. Our work is motivated by the Versioning machine, as we build a tool MultiVersioner that visualizes several documents at once, and provides the user with a rich set of relevant information regarding the comparison of the versions.

Our primary user is Tanya Clement who is a graduate student at the UMD English Department, who is also affiliated with the Maryland Institute of Technology in the Humanities (MITH). As a part of her research, Ms. Clement compares several versions of poems. Ms. Clement uses Versioning Machine to aid her in comparing different versions of a poem. Versioning Machine does not provide any search capabilities, both across versions as well as across documents. Also, Versioning Machine supports the display of the versions of just one document at any given time. To open another document, all versions of the current document have to be closed first. We intended to develop a tool that displayed not only versions of a single document, but also facilitate the exploration and analysis of multiple versions of multiple documents.

The remainder of the report is organized as follows. Section 2 discusses other work related to our topic. Section 3 describes our program in detail, along with various aspects associated with it. Section 4 covers the qualitative evaluation of our program. We offer directions for future work and conclude our report in section 5.

II. Related work

As mentioned earlier, Versioning Machine developed by Schreibman et all is the tool that motivated our work. The paper [1] contains a detailed description of the mechanics of Versioning Machine and how the tool facilitates comparing different versions of a document. They require the data to be encoded in XML format. They enable the display of several versions of a document on the screen, and also denote the additions and deletions of one version with respect to others.

Viégas et all [2] address the issue of monitoring history changes in wikipedia articles. Their work overlaps to some extent with our project and provided us with some insight on representing change in documents. Also relevant is the discussion of movement of text within a document, particularly when content is added or deleted from certain versions of a document. [4] covers the aspects of movement in text extensively. An important aspect of our tool was highlighting the similarities and differences across several documents. Very little literature was available on this topic, but we were able to borrow a few pointers from *ScentHighlights* by Chi et all [5]. ScentHighlights is a tool developed to highlight results based on search words provided by the user. This high-level idea of using highlighting to display search results was incorporated into MultiVersioner.

Plagiarism detection and source code comparison are areas that have been extensively researched, we reviewed plenty of literature and tools in this area. *CHECK* [7], which concentrated on the algorithmic approaches to document comparison, is a representative of the work in the area of plagiarism detection. Brin et all [8] describe copy detection mechanisms in digital content. The Visual Code Navigator by Lommerse et all [10] is a recent work that focuses on source code investigation.

BasketLens [11], *FeatureLens* [12], and *Emily* [13] are all projects developed at the University of Maryland, which are very closely related to our project. Emily is a tool designed to visualize and analyze unstructured human-generated text like poetry, in particular poetry by Emily Dickinson. BasketLens, which was based on Emily, explored the eroticism in Emily Dickinson's poem. BasketLens introduced the concept of baskets, defined as "a group of words unified by some concept".

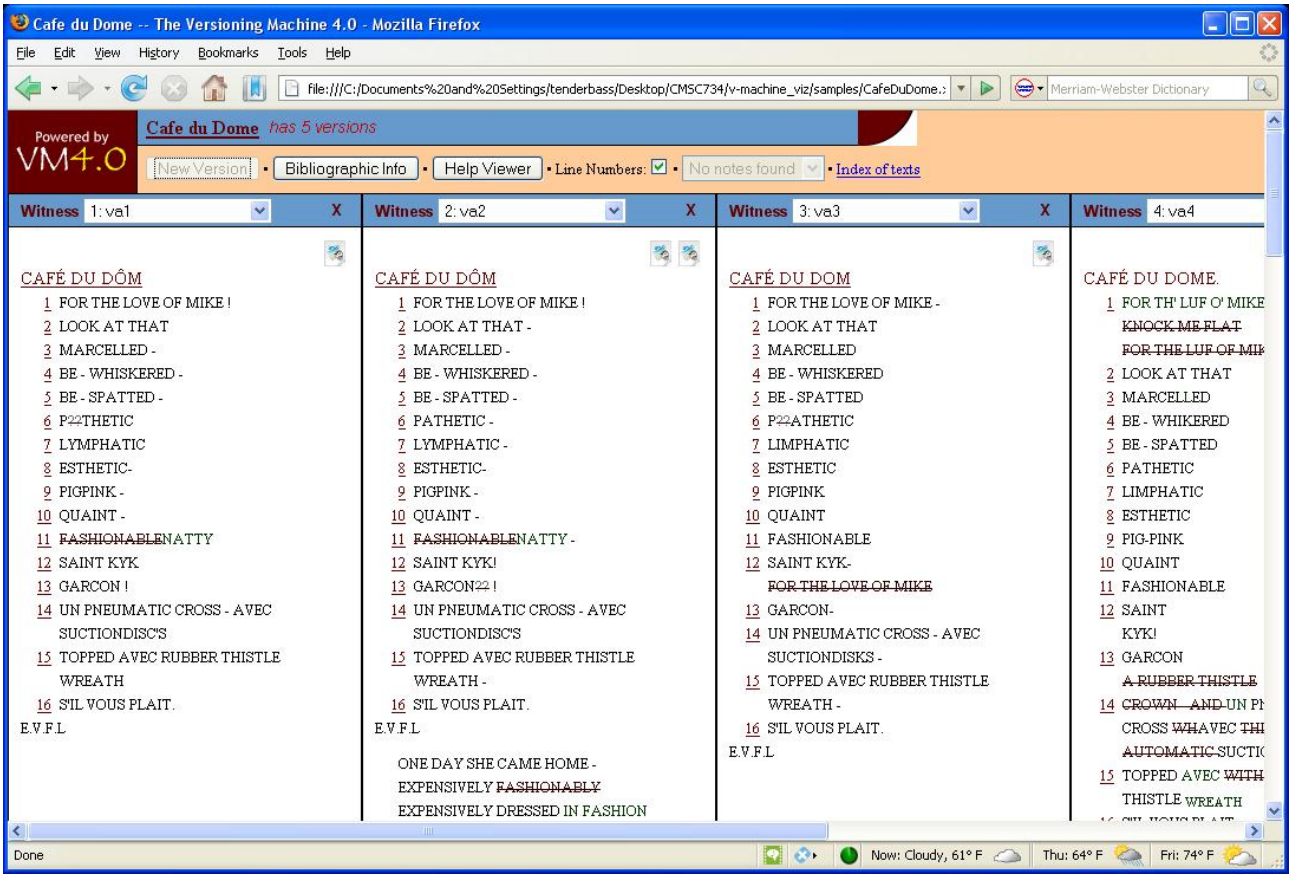

Figure 1. Versioning Machine 4.0

An example of a basket could be the category *flowers*. A search performed on this basket would return instances of *rose*, *daisy* etc. *FeatureLens* was a class project for Information Visualization in Spring 2006, that facilitated pattern finding in text collections by providing visualizations of the results of text mining algorithms.

The following sources covering eclectic topics were also of interest to us. Hongyuan Zha and Xiang Ji [9] adopt a novel approach to compare multilingual documents, by making use of bipartite graphs. Veksler et all [6] try to predict where the human eye is likely to catch information, and provide suggestions to make use of their predictions in order to aid viewing of information.

III. Description of the Program

A. Philosophy

The goal of our project was two-fold, to provide an effective overview of the content and size of all documents, as well as to provide a detailed display of versions of one document, along with a variety of search capabilities. Our initial aim was to provide an overview of the contents of various documents, as well as versions of a single document. This was in accordance with the Shneiderman Mantra *Overview first, zoom and filter, details on demand*. At a very high level, we decided not to make an explicit distinction between versions of a single document vs different documents; we decided to treat documents, as well as versions of a single document,

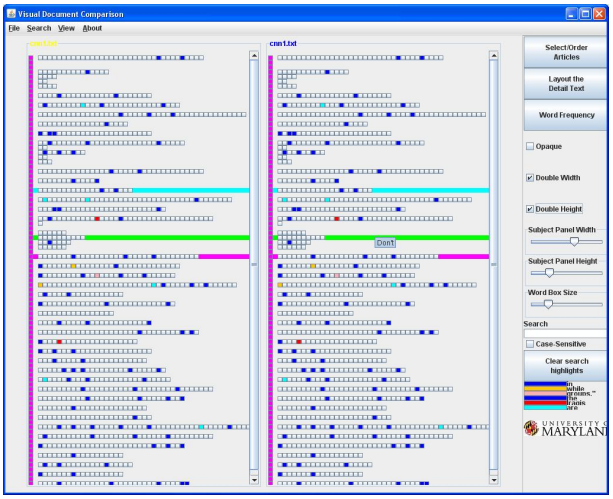

Figure 3. Viewing two long versions

alike. When a document is opened, it is displayed in a *version panel*. Double clicking the version panel brings up a window containing the document text. For example, the user could open up five versions of the poem *Autumn*, and search for the word *withers*. Based on the results, the user may then want to search for *withers* in a different poem *Nocturne*. The user just has to open up a version of poem *Nocturne*, and perform the same search again, to notice instances of the word withers in

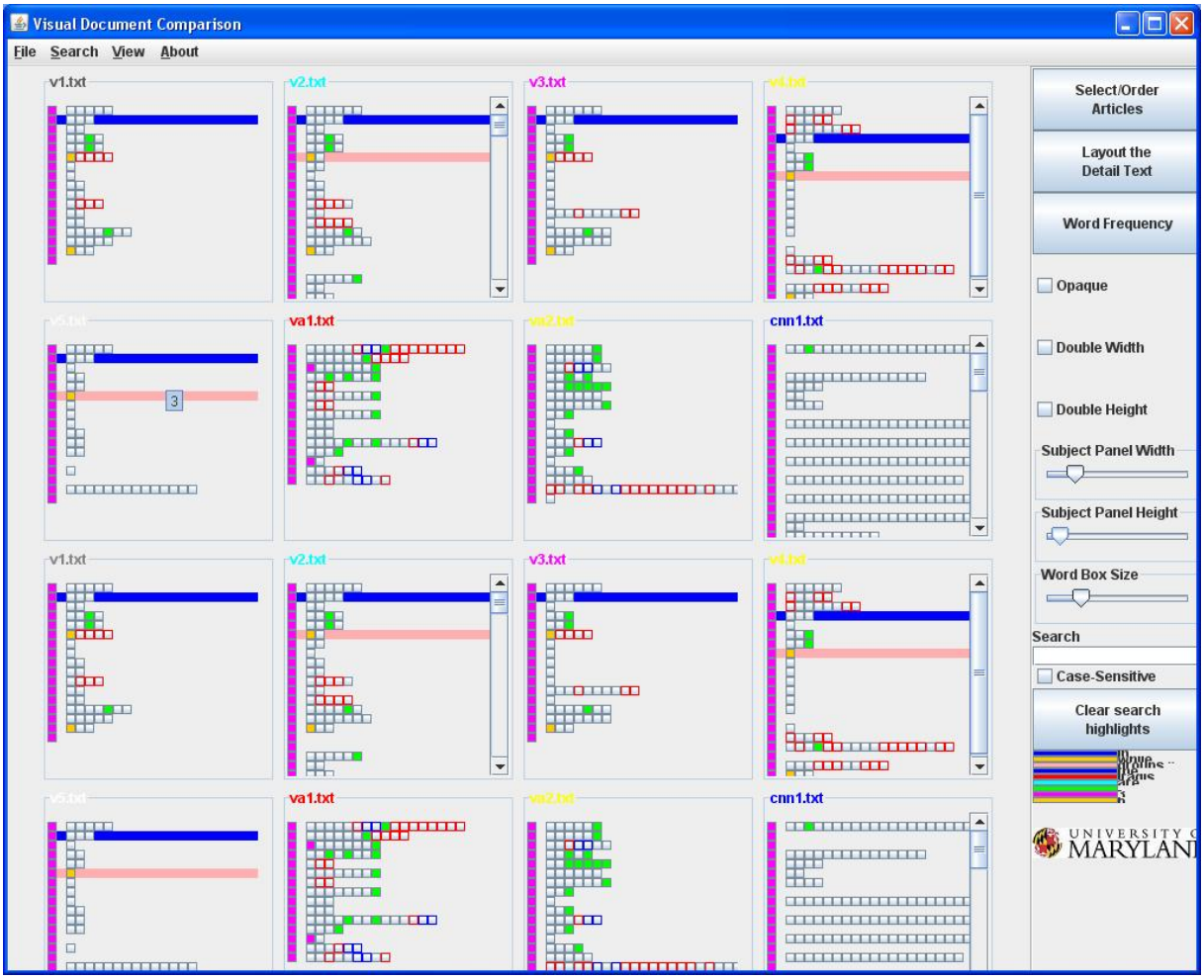

Figure 2. Overview of many versions and documents in Multiversioner

Nocturne.

B. Description of the interface

We started out by building the interface to display the overview of documents in MultiVersioner. In the default screen, words are denoted by equally sized boxes. Mousing over a box pops up a tooltip containing the entire sentence, with the current word being shown in bold. In the tooltip, words added in the current version, that are not present in other versions, are displayed in italics. Words that are missing in the current Verizon, but are present in other versions, are struck through. All the controls and buttons are located at the right of the interface.

C. Text View

Using word boxes to represent words is used primarily to obtain an overview of all the documents. To perform detailed anaysis on certain versions, a representation displaying the actual sentences, instead of word boxes, is preferred. Switching to the text representation can be done by choosing the *Fit Text* option under the *View* menu. All the functionalities described using word boxes like word searches, line searches, displaying the frequency table, etc are available under this display as well.

D. Detail window

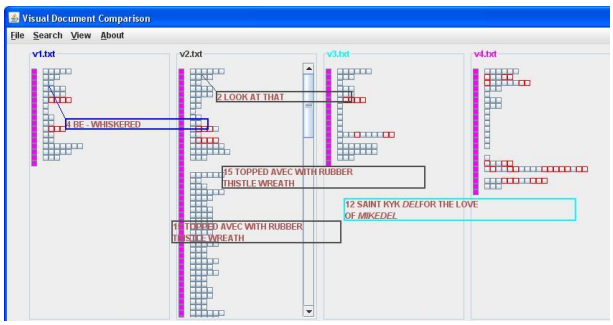

Figure 4. Pop-up Detail Window

Clicking on a box brings up a *detail window* (Figure 4) containing the entire sentence. The purpose of the detail window is to display a sentence of interest on the screen, analogous to a post-it note. Detail windows can be either opaque or transparent, in the situation where a detail window overlaps one or more version panels. If a detail window is opaque, the contents of the detail window are displayed, while possibly occluding some of the word boxes in the version panels. If a detail window is transparent, the version panels

will be visible, but the text in the detail window may be occluded. Detail windows are transparent by default, the user has the option of making it opaque by checking the *Opaque* checkbox. The button *Layout Detail Windows* arranges the detail windows in an orderly manner. It should be noted that the detail windows are colored using the same border color of their originating version panel. Furthermore, if the user still needs to find out where a detail window originated from, he could double click the detail window, which causes the corresponding boxes in the originating version panel to flicker. The converse of the above action could be performed as well, i.e. double clicking on a particular box would cause its corresponding detail window (if available) to flicker. A detail window could be closed in a couple of ways. First, the user could right click on the detail window and choose the *close* option. The other way is to drag the detail window away from the screen boundary, i.e. *causing it to go out of the screen.*

E. Search

MultiVersioner provides extensive search features to facilitate comparison between several documents. The very basic search feature is the word search. A search bar is provided, where the user can type in a word to be search across all documents that are open. By default, searches are not case-sensitive, but a checkbox is provided if the user wants the search to be case-sensitive. An alternative way to perform a word search is to right click the word box to be searched. Search results are color coded, and appearances of the search word in all documents are colored using the same color. Different colors are used for different searches. We tried to ensure the use of different hues to represent different colors, so as to provide appropriate contrast between search results. A search history is available at the bottom of the panel in the right. A button to clear the searches across the version panels is available as well. Even if the searches are cleared across the version panels, the search history will remain on the screen.

A line search feature is available as well. Each line contains an magenta-colored anchor which. Right clicking the anchor triggers a line search, where the specified line will be searched across all documents. Our line search algorithm ,which works by comparing the number of matching words between pairs of lines, returns only matches that are reasonably relevant to the specified line. As with the word search, the line search colors matching lines throughout all the documents using the same color.

F. Word Frequency Table

Multiversioner computes a frequency table containing statistical information about the words present in all documents and their versions. When comparing different versions of a document or comparing different documents that are related, researchers in literature have a need to identify unique and common words and sentences. We believe that an approach as simple as a frequency table listing is powerful in providing insight. Given the number of articles, and the number of times that a word appears in each article, users can know which words are common across documents and which ones are unique to a single document. Furthermore, the results exported in Excel/CSV format can help users use the information as they wish.

G. Other features

This section wraps up the discussion of the program interface by detailing the various other features available in MultiVersioner. It should be mentioned here that double clicking the version panel brings up a text file containing the entire contents of the document. There are sliders available to control the version panel height, width and the sizes of the word boxes. There are checkboxes which double the width and height of the version panels as well. We also have a scroll lock that is functional if the any of the version panels contain long documents that require scroll bars. Choosing the scroll lock by checking the *Scroll Lock* checkbox synchronizes the scrollable documents with each other. If one document is scrolled, other documents scroll as well. We conclude this section by noting that a brief user manual is available under the *About* menu, outlining the functionalities of all the controls of MultiVersioner.

H. An Example Scenario

Here is an example of the proceedings of a typical Multiversioner session.

1) The user, let us call her Sarah, wishes to glance through the articles that are present. Sarah uses the mouse throughout the interaction.
2) Depending on the size of the articles and the screen, Sarah may resize the version panels to utilize a large portion of the screen space.
3) Sarah may want to search for a particular word, and would do so by clicking the right mouse button to trigger a search on that word. The word Box corresponding to the searched keywords are highlighted in different colors in each search. Also, the legend of color and corresponding search keywords are shown in the bottom of the tool panel on the right.
4) Sarah can then perform a line search. She can use the right mouse button clicking on the anchor box, which is a magenta colored box present at the beginning of a line. Then each sentence similar to the selected sentence will be highlighted in the same color, just like a word search.
5) If Sarah finds any sentence that is of interest, she clicks on a word box to retain that sentence on the screen in a manner analogous to post-it notes. In order to relate a detail window with the word box it originated from, the color of detail window's border is the same as that of the corresponding version panel, and a line is drawn from the detail window to its originating word box.
6) After Sarah has finished performing a fair amount of exploration and has created a certain number of detail windows that are of interest to her, she has the option of choosing the *Layout the detail texts* button to automatically arrange all the detail windows.

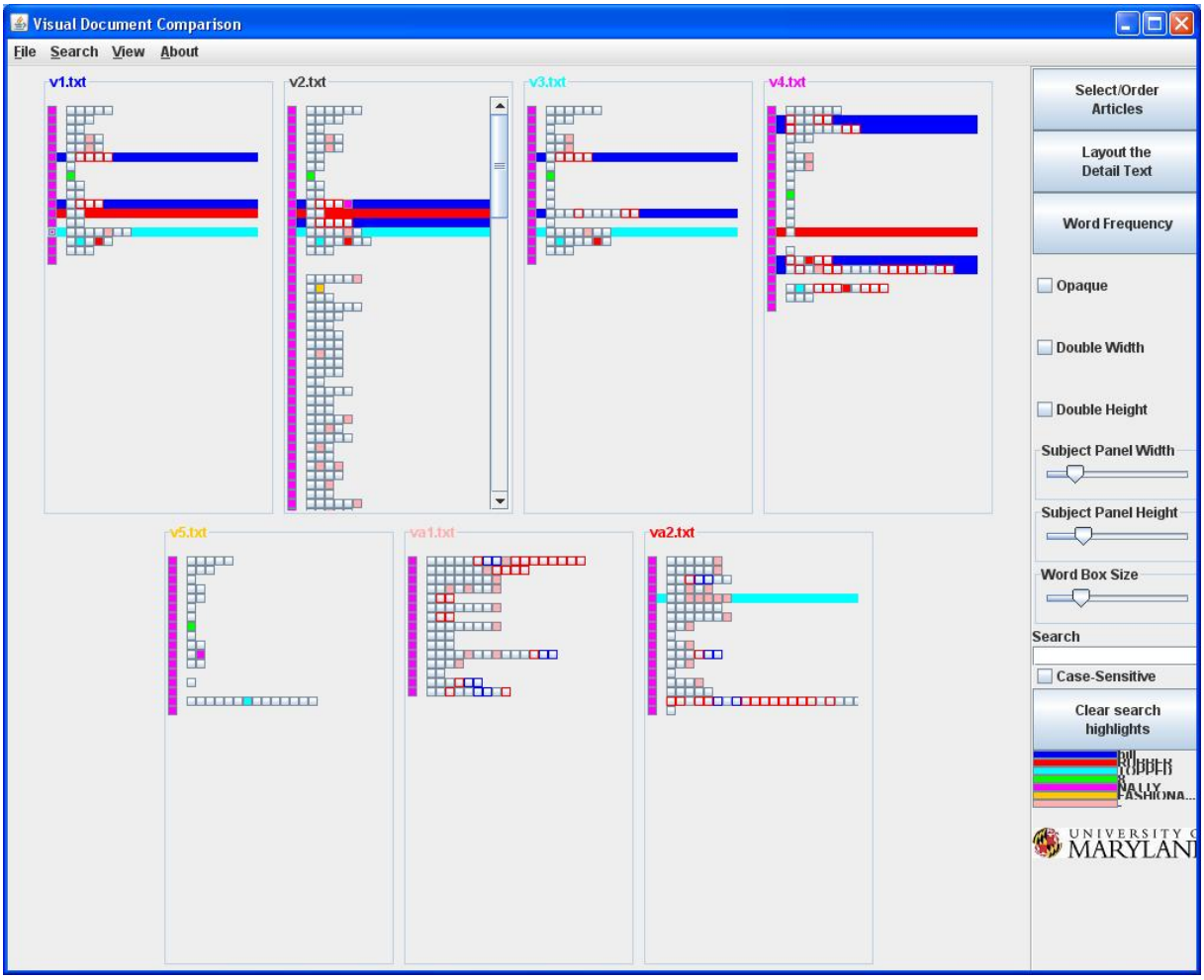

Figure 5. Highlighting word and line search results

7) If she want to see the results of the search made in step 4 and 5 (doing so is currently not convenient as the detail windows are opaque, and occlude the word boxes of some version panels), she can use make the detail window transparent by unchecking the *Opaque* checkbox.
8) If Sarah is viewing long documents, she has the option to synchronize the vertical scroll bars, so that scrolling one document would automatically cause the other documents to scroll as well. The advantage of this feature is that all the versions would display the same part. The above functionality can be turned on by checking the *Scroll lock* checkbox.
9) If she finds it difficult to associate the words with the boxes, she has the option of choosing the *Fit Text* option under the *View* menu so that actual words are displayed as against boxes.

IV. Evaluation

The main aim of this evaluation was to improve our tool by enhancing its utility for our target users. The evaluation was a continuous process during which, the input received from participants was incorporated into the tool and further feedback was obtained. This process was repeated several times. We received inputs from two types of users: 1) Domain expert for whom the tool was aimed at and 2) Domain novices.

A. Domain Expert

As mentioned earlier, Tanya Clement was our domain expert. We started by understanding her requirements and this was followed up by several evaluation sessions. During these sessions, the protocol followed was first performing a demonstration of the new features implemented, followed by usability testing and feedback regarding the advantages and disadvantages of Multiversioner with respect to the Versioning Machine. Since we were working with only one user, the nature of our evaluation was qualitative.

1) Document layout: Ms. Clement wanted all versions of a single document to be readily distinguishable from others. After brainstorming several possibilities with her, we decided that coloring borders of the version panels belonging to the same document with the same color was appropriate. She also wanted to have the ability to move the version panels around by dragging them, and placing them at a location of her convenience as this would enable her to place two versions of interest together and make observations.

2) Search: The ability to search for words was much appreciated by Tanya, but she seemed slightly confused by the line search results as our algorithm did not classify certain results as matches. On being explained how the algorithm works she seemed satisfied and gave us some of her requirements which we are incorporating to improve the search results.

3) Frequency Table and Statistical Analysis: One of the features she required was displaying the unique words in each version. To incorporate this, we construct a frequency table which enables users to view the frequency of occurrence of words across different document versions. From the frequency table, words unique to a particular version are identified and highlighted. In the next stage she suggested that a user be allowed to see the spatial occurrences of a particular word by clicking it in the frequency table. This feature was also added to our tool. The frequency table forms the basis for developing several statistical analysis and data mining techniques to aid the analysis. These techniques can be added at a later stage.

4) Word Boxes vs Text: Ms. Clement stressed that she prefers seeing the actual words, rather than they being represented as word boxes. If that is the case, then she said she does not have to think back and forth, attempting to associate boxes with words. She added that boxes were helpful for a high level view consisting of a large number of documents and versions, but text was more useful for her analysis.

5) Miscellaneous features: Among all the other features, she found the synchronized scrolling helpful. She also suggested that we link the detail window to its originating location in the version panel by drawing a line. On implementation, this did not work very well as one tends to confuse between the intersecting lines. Following this, we implemented the feature where double clicking the detail window causes the originating version panel to flicker, which she found quite helpful. She did mention that once the originating source was identified, the flicker should be disabled by clicking on the source. She was amused by the feature of making the detail window disappear by drag-and-dropping it out of the screen.

B. Domain Novices

Here we summarize the feedback and suggestions received from people who were not experts in English literature and poetry. The evaluation method primarily consisted of a demonstration, with the participants occasionally playing with the controls and then telling us what they liked or disliked. Most of the input we received was specific to visualization aspects such as layout and color schemes.

During the entire course of this project we got regular feedback from Dr. Shneiderman. Initially our main focus was on providing an overview by using equal-sized word boxes to denote the words of a sentence. Dr. Shneiderman suggested that users across most domains would prefer the actual text being displayed, rather than seeing the word boxes. In our final version, we were able to capture the flavor of Dr. Shneiderman's suggestion, but are facing minor implementation issues, which we hope to resolve. His other suggestions included the use of a gray background, rather than a black background, which would distract the user's attention from other aspects of the interface.

We also performed a live demonstration before our fellow CMSC 734 classmates, all of whom have a reasonable knowledge of several advanced visualization concepts, and received additional feedback from them. They liked the idea of searching for words across versions and synchronized scrolling across document versions. Feedback included associating the detailed windows with their position in the documents, using less contrasting colors for the document background and allowing users to see the words instead of boxes. One suggestion was the effective utilization of the entire screen space to display the documents. This prompted us to implement sliders enabling the users to manually control the height and width of the documents. We also offer additional suggestions for optimizing the screen space utilization.

C. Common Feedback

There were a couple of suggestions which were offered by both user groups. The first suggestion was the preference of seeing actual text in the version panels as against the word boxes. They reported that correlating the word boxes to the poems themselves is confusing on occasions. As we received this comment from several sources, we appreciated the importance of displaying the actual text. Also, we had initially failed to relate the detail windows with the word boxes they originated from; both user groups pointed out that we needed to have a mechanism to associate the detail windows with their originating word boxes.

V. FUTURE WORK AND CONCLUSIONS

There exist several tools for comparing documents, articles and software code based on some similarity metric and a few of these possess the functionality to enable the user to perform statistical analysis on the documents [11][12]. A related area that has not been extensively researched is facilitating the comparison of multiple versions of a document. Versioning Machine is probably one of the first tools that enables this. We built upon Versioning Machine in the following two ways:

- Allowing the user to compare multiple documents, each of which consists of multiple versions
- Add the ability to search for entities such as words and lines across the documents and versions, and analyze the frequency patterns of these entities

In the process, we also explored visualization techniques and gained some important insights. We looked into using color to perform effective highlighting and for discovering patterns. The aspect of providing an overview, with details available on demand, was implemented, which helped us appreciate some of the issues involved. We have demonstrated the effectiveness of our tool for comparing poems, but the underlying concepts are applicable to several other domains such as comparison of patents or in discovering the evolution of laws over time. The search and comparison methods can be suitably modified to discover meaningful relationships between the respective entities.

As MultiVersioner was developed as a part of a class project, we were able to implement only a limited number of features. We now outline the various possibilities that can be explored

in the context of visualizing multiple document versions. As our primary domain of focus was poems, MultiVersioner works best for data of small sizes. Future work could involve addressing the problem of visualizing long documents, which could for example be used in the contexts of comparing different versions of programs or different versions of wikipedia articles. The possibility of utilizing the entire screen space to fit all open documents should be examined as well. In this case, opening new documents or closing existing ones, should result in the program dynamically resizing the documents into equal segments that fit the screen. While doing this one would also like to restrict changes in configuration of the layout and ensure that the users do not lose context. Other areas that can be studied are using color for grouping and highlighting hierarchical entities like documents, paragraphs, lines and words. Adding additional features inspired by [11], [12] and adapting them for the purpose of comparing document versions to aid the statistical analysis would substantially increase the utility of Multiversioner. There is ample potential to research effective interfaces in this area, and we explored some of the possibilities here by implementing MultiVersioner.

REFERENCES

[1] Susan Schreibman, Amit Kumar and Jarom McDonald. (2003). The Versioning Machine, *Literary and Linguistic Computing,* 18(1), 101-107

[2] Fernanda B. Viégas, Martin Wattenberg, Kushal Dave. (2004). Studying cooperation and conflict between authors with history flow visualizations, *Proceedings of the SIGCHI conference on Human factors in computing systems*, Vienna, Austria, 575-582.

[3] Nancy E. Miller, Pak Chung Wong, Mary Brewster, Harlan Foote. (1998). TOPIC ISLANDS TM – A Wavelet-Based Text Visualization System, *Proceedings on the conference of visualization*, North Carolina, 189-196.

[4] Kenton O'Hara, Abigail Sellen. (1997). A Comparison of Reading Paper and On-Line Documents, *Proceedings of the SIGCHI conference on Human factors in computing systems*, Georgia, 335-342.

[5] Ed H. Chi, Lichan Hong, Michelle Gumbrecht, Stuart K. Card. (2005). ScentHighlights: highlighting conceptually-related sentences during reading, *Proceedings of the 10th international conference on Intelligent user interfaces*, California, 272-274.

[6] Vladislav Daniel Veksler, Wayne D. Gray. (2007). Mapping semantic relevancy of information displays, *Proceedings of ACM CHI 2007 Conference on Human Factors in Computing Systems* v.2 2729-2734

[7] Antonio Si, Hong Va Leong, Rynson W. H. Lau. (1997). CHECK: a document plagiarism detection system, *Proceedings of the 1997 ACM symposium on Applied computing*, California, 70-77.

[8] Sergey Brin, James Davis, Hector Garcia-Molina. (1995). Copy detection mechanisms for digital documents, *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, California, 398-409.

[9] Hongyuan Zha, Xiang Ji. (2002). Correlating multilingual documents via bipartite graph modeling, *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, Finland.

[10] G. Lommerse, F. Nossin, L. Voinea, A.Telea. (2005). The Visual Code Navigator: An Interactive Toolset for Source Code Investigation, *in Proc. IEEE InfoVis'05, IEEE CS Press*, 24-31.

[11] Darya Filippova. (2007). BasketLens: interface for document visualization and exploration, *Independent study conducted with Ben Shneiderman and Catherine Plaisant.*

[12] Don, A., Zheleva, E., Gregory, M., Tarkan, S., Auvil, L., Clement, T., Shneiderman, B., Plaisant, C. (2007). Discovering interesting usage patterns in text collections: integrating text mining with visualization, *HCIL Technical report.*

[13] Nitin Madnani. (2005). Emily: A Tool for Visual Poetry Analysis.